

## Задача А. Снеговики

Имя входного файла: `snowmen.in`  
Имя выходного файла: `snowmen.out`  
Ограничение по времени: 4 секунды  
Ограничение по памяти: 64 мегабайта

Зима. 2012 год. На фоне грядущего Апокалипсиса и конца света незамеченной прошла новость об очередном прорыве в областях клонирования и снеговиков: клонирования снеговиков. Вы конечно знаете, но мы вам напомним, что снеговик состоит из нуля или более вертикально поставленных друг на друга шаров, а клонирование — это процесс создания идентичной копии (клона).

В местечке Местячково учитель Андрей Сергеевич Учитель купил через интернет-магазин «Интернет-магазин аппаратов клонирования» аппарат для клонирования снеговиков. Теперь дети могут играть и даже играют во дворе в следующую игру. Время от времени один из них выбирает понравившегося снеговика, клонирует его и:

- либо добавляет ему сверху один шар;
- либо удаляет из него верхний шар (если снеговик не пустой).

Учитель Андрей Сергеевич Учитель записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

### Формат входных данных

Первая строка содержит количество действий  $n$  ( $1 \leq n \leq 200\,000$ ). В строке номер  $i + 1$  содержится описание действия  $i$ :

- $t\ m$  — клонировать снеговика номер  $t$  ( $0 \leq t < i$ ) и добавить сверху шар массой  $m$  ( $0 < m \leq 1000$ );
- $t\ 0$  — клонировать снеговика номер  $t$  ( $0 \leq t < i$ ) и удалить верхний шар. Гарантируется, что снеговик  $t$  не пустой.

В результате действия  $i$ , описанного в строке  $i + 1$  создается снеговик номер  $i$ . Изначально имеется пустой снеговик с номером ноль.

Все числа во входном файле целые.

### Формат выходных данных

Выведите суммарную массу построенных снеговиков.

### Примеры

snowmen.in	snowmen.out
8	74
0 1	
1 5	
2 4	
3 2	
4 3	
5 0	
6 6	
1 0	

## Задача В. Вперёд!

Имя входного файла: `movetofront.in`  
Имя выходного файла: `movetofront.out`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайт

Капрал Дукар любит раздавать приказы своей роте. Самый любимый его приказ — «Вперёд!». Капрал строит солдат в ряд и отдаёт некоторое количество приказов, каждый из которых звучит так: «Рядовые с  $l_i$  по  $l_j$  — вперёд!»

Перед тем, как Дукар отдал первый приказ, солдаты были пронумерованы от 1 до  $n$  слева направо. Услышав приказ «Рядовые с  $l_i$  по  $l_j$  — вперёд!», солдаты, стоящие на местах с  $l_i$  по  $l_j$  включительно, продвигаются в начало ряда в том же порядке, в котором были.

Например, если в какой-то момент солдаты стоят в порядке 2, 3, 6, 1, 5, 4, то после приказа «Рядовые с 2 по 4 — вперёд!», порядок будет таким: 3, 6, 1, 2, 5, 4. А если потом Капрал вышлет вперёд солдат с 3 по 4, то порядок будет уже таким: 1, 2, 3, 6, 5, 4.

Вам дана последовательность приказов Капрала. Найдите порядок, в котором будут стоять солдаты после исполнения всех приказов.

### Формат входных данных

В первой строке входного файла указаны числа  $n$  и  $m$  ( $2 \leq n \leq 100\,000$ ,  $1 \leq m \leq 100\,000$ ) — число солдат и число приказов. Следующие  $m$  строк содержат приказы в виде двух целых чисел:  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq n$ ).

### Формат выходных данных

Выведите в выходной файл  $n$  целых чисел — порядок, в котором будут стоять солдаты после исполнения всех приказов.

### Примеры

<code>movetofront.in</code>	<code>movetofront.out</code>
6 3 2 4 3 5 2 2	1 4 5 2 3 6

## Задача С. Перестановки strike back

Имя входного файла: `permutation2.in`  
Имя выходного файла: `permutation2.out`  
Ограничение по времени: 5 секунд  
Ограничение по памяти: 256 мегабайт

Вася выписал на доске в каком-то порядке все числа от 1 по  $N$ , каждое число ровно по одному разу. Иногда он стирает какое-то число и записывает на его место другое. Количество чисел, выписанных Васей, оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая в любой момент отвечает на вопрос — сколько среди чисел, стоящих на позициях с  $x$  по  $y$ , по величине лежат в интервале от  $k$  до  $l$ . Сделайте то же самое.

### Формат входных данных

В первой строке лежит два натуральных числа —  $1 \leq N \leq 100\,000$  — количество чисел, которые выписал Вася и  $1 \leq M \leq 100\,000$  — суммарное количество вопросов и изменений сделанных Васей. Во второй строке дано  $N$  чисел — последовательность чисел, выписанных Васей. Далее в  $M$  строках находятся описания вопросов. Каждый запрос на изменение числа в некоторой позиции начинается со слова **SET** и имеет вид **SET a b** ( $1 \leq a \leq N$ ,  $1 \leq b \leq N$ ). Это означает, что Вася изменил число, записанное в позиции  $a$  на число  $b$ . Каждый Васин вопрос начинается со слова **GET** и имеет вид **GET x y k l** ( $1 \leq x \leq y \leq N$ ,  $1 \leq k \leq l \leq N$ ).

### Формат выходных данных

Для каждого Васиного вопроса выведите единственное число — ответ на Васин вопрос.

### Примеры

<code>permutation2.in</code>	<code>permutation2.out</code>
4 4	1
1 2 3 4	3
GET 1 2 2 3	2
GET 1 3 1 3	
SET 1 4	
GET 1 3 1 3	

## Задача D. Откат

Имя входного файла: `rollback.in`  
Имя выходного файла: `rollback.out`  
Ограничение по времени: 1.5 секунды  
Ограничение по памяти: 256 мегабайта

Сергей работает системным администратором в очень крупной компании. Естественно, в круг его обязанностей входит резервное копирование информации, хранящейся на различных серверах и «откат» к предыдущей версии в случае возникновения проблем.

В данный момент Сергей борется с проблемой недостатка места для хранения информации для восстановления. Он решил перенести часть информации на новые сервера. К сожалению, если что-то случится во время переноса, он не сможет произвести откат, поэтому процедура переноса должна быть тщательно спланирована.

На данный момент у Сергея хранятся  $n$  точек восстановления различных серверов, пронумерованных от 1 до  $n$ . Точка восстановления с номером  $i$  позволяет произвести откат для сервера  $a_i$ . Сергей решил разбить перенос на этапы, при этом на каждом этапе в случае возникновения проблем будут доступны точки восстановления с номерами  $l, l+1, \dots, r$  для некоторых  $l$  и  $r$ .

Для того, чтобы спланировать перенос данных оптимальным образом, Сергею необходимо научиться отвечать на запросы: для заданного  $l$ , при каком минимальном  $r$  в процессе переноса будут доступны точки восстановления не менее чем  $k$  различных серверов.

Помогите Сергею.

### Формат входных данных

Первая строка входного файла содержит два целых числа  $n$  и  $m$ , разделенные пробелами — количество точек восстановления и количество серверов ( $1 \leq n, m \leq 100\,000$ ). Вторая строка содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  — номера серверов, которым соответствуют точки восстановления ( $1 \leq a_i \leq m$ ).

Третья строка входного файла содержит  $q$  — количество запросов, которые необходимо обработать ( $1 \leq q \leq 100\,000$ ). В процессе обработки запросов необходимо поддерживать число  $p$ , изначально оно равно 0. Каждый запрос задается парой чисел  $x_i$  и  $y_i$ , используйте их для получения данных запроса следующим образом:  $l_i = ((x_i + p) \bmod n) + 1$ ,  $k_i = ((y_i + p) \bmod m) + 1$  ( $1 \leq l_i, x_i \leq n$ ,  $1 \leq k_i, y_i \leq m$ ). Пусть ответ на  $i$ -й запрос равен  $r$ . После выполнения этого запроса, следует присвоить  $p$  значение  $r$ .

### Формат выходных данных

На каждый запрос выведите одно число — искомое минимальное  $r$ , либо 0, если такого  $r$  не существует.

### Примеры

rollback.in	rollback.out
7 3	1
1 2 1 3 1 2 1	4
4	0
7 3	6
7 1	
7 1	
2 2	